Detailed Tuning and Validation of Hardware Simulators through Microbenchmarks

Rommel Sánchez Verdejo*†, Petar Radojković*
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {rommel.sanchez, petar.radojkovic}@bsc.es

Keywords—Simulation, Memory simulation, CPU Simulation, DRAM, DRAM Simulation, x86

I. Extended Abstract

Hardware simulators are used by the academia and industry to prototype, explore and evaluate novel microarchitectural features. Because of its importance, it is imperative to pay special attention to their validation. Unfortunately, this process is not standardized. In this work, we describe a set of microbenchmarks for the validation of the CPU execution units and the memory subsystem, including on-chip caches and main memory. Also, we present a case study in which the microbenchmarks are used to validate a simulation infrastructure based on the ZSim [1] and DRAMSim2 [2] vs. an actual Sandy Bridge server. The presented case study shows how the microbenchmarks can be used to isolate the resource behavior of the target architecture and pinpoint the specific differences between the simulator and the target hardware.

A. Microbenchmarks: CPU execution units

All the microbenchmarks are designed using the principle that is presented in Table I. Each benchmark consist of four parts: (1) The register used as a loop iteration counter (`ecx`) is set for 10,000 times of execution; (2) The main section of the benchmark is a sequence of single repetitive instruction of the target ISA; (3) The sequence of target instructions is followed with the decrement of the loop counter register; (4) Finally, the counter value is compared with zero followed by the conditional branch to the beginning of the loop.

B. Microbenchmarks: Caches and main memory

The benchmarks that stress the caches and memory are implemented using the concept of pointer chasing. In the benchmark prologue, we allocate a contiguous section of memory and initialize it to a given array element that contains the address of the next element to fetch. The benchmarks are initialized to (1) Traverse the whole array; (2) Access different cache lines in each memory access; (3) Memory accesses have a random pattern, preventing data prefetchers to bring data to any level of cache.

Table II shows the code for memory latency microbenchmarks, which behavior are explain as follows (1) The register used as a loop iteration counter (`ecx`) is initialized; (2) The initial address of the array is passed to the assembly code as an input parameter; (2) The main part of the benchmark is a sequence of indirect load instructions (`mov(%rax), %rax`)

Line	Source code	Explanation
00001	`mov $10000, %ecx`	Initialize loop counter *ecx* to 10,000
00002	`start_loop:`	beginning of the loop
00003	`ADC %eax, %ebx`	target instruction
00004	`ADC %eax, %ebx`	target instruction
...	`...`	...
10002	`ADC %eax, %ebx`	target instruction
10003	`dec %ecx`	decrement loop counter
10004	`jnz start_loop`	if (counter $\neq$ 0) jump to start_loop

TABLE I. Structure of microbenchmark ASM core.

Line	Source code	Explanation
0001	`register struct line *next asm("rax");`	strcut line owns pointer to the next accesss
0002	`register int i asm("ecx");`	`ecx` is the loop counter
0003	`i = 1000000;`	`C` initialization of the loop counter
0004	`next = ptr->next;`	First memory access in `C` form
0005	`start_loop:`	beginning of the loop
0006	`mov (%rax), %rax`	load instruction (pointer chasing)
0007	`mov (%rax), %rax`	load instruction (pointer chasing)
...	`...`	...
1007	`mov (%rax), %rax`	load instruction (pointer chasing)
1008	`dec %ecx`	decrement loop counter
1009	`jnz start_loop`	if (counter $\neq$ 0) jump to start_loop

TABLE II. Structure of memory latency microbenchmark.

that traverse the memory access pattern; (3) The sequence of target instructions is finalized with the decrement of the loop counter register and an exit condition or jump to the beginning of the iteration. The assembly loop is wrapped-up by the `C` program which reads a previously generated file containing information about the array size and the random access pattern.

C. Case study: ZSim and DRAMSim2 vs. Intel Xeon E5-2670 SandyBridge-EP

We performed a case study in which the microbenchmarks are used to compare the simulation infrastructure integrated with ZSim and DRAMSim2 simulators. ZSim is an execution-driven CPU simulator widely used in the computer architecture research, developed to mimic the Westmere architecture and validated against real hardware using the SPEC CPU 2006 benchmark suite. DRAMSim2 is a cycle accurate model of a DRAM memory controller, DIMMs, and buses by which they communicate. It is validated against DRAM manufacturer's Verilog models.

We validate the simulators vs. a dual-socket platform. Each socket with an Intel Xeon E5-2670 SandyBridge-EP processor [3] operating at 3.0 GHz. The main memory is 16 GB and is connected to the processors using four DDR3-1600 channels. Each processor runs eight cores, the hyper-threading

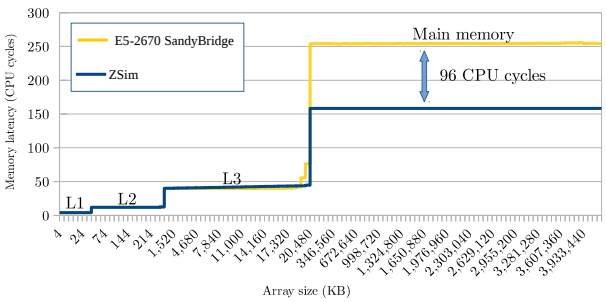

Fig. 1. Memory access latency: L1, L2, L3 cache and main memory.

feature has been disabled like in most HPC systems [4].

D. Case study: CPU Execution units

We automated the creation of 346 microbenchmarks covering a wide range of logic for integer and floating point instructions. Then, we executed the benchmarks in actual hardware and the simulator. Out of 346 tested instructions, the simulator matches around 55% the actual system latency while in 28% the simulation error is moderate or high meaning that exceeds 50% of the CPI. Since each of the microbenchmarks stresses one resource at a time, finding the sources of the simulation error is relativeley simple. The enhanced version of the simulator show much better accuracy 74% of all the instructions correctly match the actual system while less than 9% of them show a simulation error of above 50% of the CPI. Moderate or high simulation error come mainly from the complicated instructions or CPI dependency on the operand values. Detailed analysis of these cases is an ongoing work.

E. Case study: Caches and main memory

To compare the caches and main memory access time between the selected simulators and the actual hardware, we used the same strategy described in Section I-B. Table III summarizes information about array sizes used in this study to stress the memory hierarchy. Results from such comparison are shown in Figure 1. The horizontal axis of the figure represents the size of the traversed array, while the vertical axis displays the memory access latency in CPU cycles. From the figure, we can distinguish four steps of the latency corresponding to the `L1`, `L2`, `L3` cache and main memory. For the cache levels, the lines overlap: ZSim cache contention accurately represents the actual system. However, for the main memory accesses, we detect a significant gap between the simulators and the real system.

These results motivated us to further explore the sources of this error. Because ZSim is a user-level simulator, it does not take into account virtual-to-physical address translation. In the real system to mitigate the address translation overheads, we used huge memory pages (1 GB per page in our study) and contiguous memory space. Simple integration of ZSim and DRAMSim2 may lead to an underestimation of the main memory access latency. ZSim simulates memory access up to the last level of cache, while DRAMSim2 is focused on the detailed timing simulation of the memory device. This implies that a direct merge of ZSim and DRAMSim2 does not consider the delay contributed by all the circuitry between the last level cache and main memory device, including the memory controller and the memory channel.

Memory Level, Size and Scope	Number of Measurements	Array sizes (range, stride)
L1 cache 32 kB, Private	8	4 kB to 32 kB , 4 kB
L2 cache 256 kB, Private	16	46 kB to 256 kB, 14 kB
L3 cache 20 MB, Shared	32	888 kB to 20 MB, 632 kB
Main memory, 16 GB/socket	64	83.69 MB to 4 GB, 63.7 MB

TABLE III. SIZE AND ORGANIZATION OF THE CACHES AND MAIN MEMORY OF THE E5-2670 SANDYBRIDGE-EP USED IN THE STUDY.

The memory latency experiments confirm that the microbenchmarks can be resourceful to detect specific errors in the simulation configurations that might be overlooked. To this date, there exist no guidelines provided by CPU simulator developers that emphasize the importance of the proper integration with the memory simulators while considering latency of the memory controller and the memory channel.

II. CONCLUSIONS

Our study provides first steps in a systematic methodology to validate computer architecture simulators. By comparing the execution of the proposed microbenchmark on both systems, we can check whether a simulator reproduces the system behavior for that particular resource. We presented a case study in which the microbenchmarks are used to validate a simulation infrastructure based on the ZSim and DRAMSim2 simulators vs. a real SandyBridge server. This study opens a discussion about the validation of the state-of-the-art-simulators used in the computer architecture community.

III. ACKNOWLEDGMENT

This work has been published in the International Conference on High Performance Computing & Simulation (HPCS) 2017, and supported by the Collaboration Agreement between Samsung Electronics Co., Ltd. and BSC, Spanish Government through Severo Ochoa programme (SEV-2015-0493), by the Spanish Ministry of Science and Technology through TIN2015-65316-P project, and by the Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272).

REFERENCES

[1] D. Sanchez and C. Kozyrakis, "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems," in *ISCA*, June 2013.

[2] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *IEEE CAL*, 2011.

[3] Intel product specification site. [Online]. Available: https://ark.intel.com/

[4] Top 500 supercomputer sites. [Online]. Available: https://www.top500.org/

Rommel Sánchez Verdejo received his M.Eng. in Computer Science from Univ. Nac. Autónoma de México, Mexico (2019). He worked for Intel Corp. in Jalisco, Mexico as a UEFI BIOS Engineer and Software Security Validation Engineer. He is pursuing a Ph.D. at Univeristat Politcnica de Catalunya jointly with the Barcelona Supercomputing Center, Spain (2106).